Journal of Computational and Applied Mathematics 27 (1989) 411–420
North-Holland

Letter Section

Note on explicit parallel multistep Runge–Kutta methods

P.J. VAN DER HOUWEN, B.P. SOMMEIJER and P.A. VAN MOURIK
Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Received 14 November 1988
Revised 24 February 1989

Abstract: This paper investigates a family of explicit two-step, two-stage Runge–Kutta methods in which the two right-hand side evaluations can be computed in parallel, so that effectively only one right-hand side evaluation per step is required. This family is compared with the family of explicit linear two-step methods of Adams type and examples of methods with increased stability intervals and methods with increased order of accuracy are given. These methods are applied to test problems taken from the test sets of Hull et al. and Enright et al., and compared with conventional linear multistep methods. In addition to the family of two-step, two-stage Runge–Kutta methods, we describe a rather general class of k-step, m-stage Runge–Kutta methods in which the m right-hand side evaluations can also be computed in parallel. For this class we indicate how the order equations and stability region can be derived.

Keywords: Numerical analysis, Runge–Kutta methods, stability, parallelism.

1. Introduction

In the literature, multistep, multistage Runge–Kutta methods (briefly MRK methods) for the initial-value problem

$$y'(t) = f(y(t)), \quad y(t_0) = y_0,$$

have been proposed in order to obtain methods with larger stability regions or with higher orders of accuracy than possessed by linear multistep (LM) methods or Runge–Kutta (RK) methods. These methods belong to a general class of integration methods, nowadays termed *general linear methods*. An excellent reference for general linear methods, and in particular MRK methods, is the recent monograph of Hairer, Nørsett and Wanner [3, p.385], where examples of and further references to special families of such methods can be found. When compared with LM methods, MRK methods require more right-hand side evaluations per step, and, when compared with RK methods, MRK methods require more storage and additional starting values.

In this note, we propose a class of explicit MRK methods which is designed in such a way that all stages in each step can be computed in parallel, so that on computers with as many processors as there are stages, the computation time is comparable with that of LM methods, that is, they require effectively only one right-hand side evaluation per step. However, the number of storage arrays may be larger. In Section 2, we analyze in some detail a family of two-step, two-stage RK methods of Adams type and we show that it is possible indeed to construct methods which have either larger stability intervals or higher orders of accuracy than is possible within the family of conventional linear two-step methods of Adams type. The starting procedure for these methods is of the same complexity as that for conventional linear two-step methods. In Section 3, the general k-step, m-stage case is defined, and in Section 4 we indicate how the order equations and the stability regions can be derived. Section 5 presents numerical results by comparing the stability and accuracy of the example methods of Section 2 with that of conventional linear two-step methods for two test problems taken from the test sets of Hull et al. [5] and Enright et al. [2]. These results justify a more thorough investigation of the general class of "parallel MRK methods" proposed in this note.

2. Two-processor algorithms

In this section, we analyze the family of two-step, two-stage RK methods of Adams type:

$$y_{n+1} = y_n + h\left[b_1 f_n + b_2 f_{n-1} + cf(a_1 y_n + a_2 y_{n-1} + b_3 h f_{n-1})\right]. \tag{2.1}$$

Here, y_n denotes a numerical approximation to $y(t_n)$, $h = t_{n+1} - t_n$, and $f_n := f(y_n)$.

If $c \neq 0$, then these methods require two right-hand side evaluations per step. For $c = 0$ the method reduces to the two-step Adams–Bashforth type method

$$y_{n+1} = y_n + h\left[b_1 f_n + b_2 f_{n-1}\right], \tag{2.2}$$

and requires only one right-hand side evaluation in each step. For all values of the parameters a_i, b_i and c the methods (2.1) are zero-stable.

The family (2.1) is chosen in such a way that it is suitable for computations on parallel computers. By writing (2.1) in the form

$$\begin{aligned} f_n &= f(y_n), \\ g_n &= f(a_1 y_n + a_2 y_{n-1} + b_3 h f_{n-1}), \\ y_{n+1} &= y_n + h\left[b_1 f_n + b_2 f_{n-1} + c g_n\right], \end{aligned} \tag{2.1'}$$

and assuming that the bulk of the numerical integration consists of the evaluations of the function f, we see that one processor can compute f_n, while at the same time the other processor can compute g_n. Hence, on two processors, the method (2.1) requires about the same computational time as the explicit linear two-step method (2.2).

The crucial point now is whether (2.1) has advantages over (2.2). For instance, do there exist in the family (2.1) methods of higher order than there are in the family (2.2) or does (2.1) contain more stable methods than (2.2)?

In order to answer these questions we need the order conditions for (2.1). It can be shown that these conditions read (cf. [4, p.196]):

$$p \geqslant 1: \quad C_0 := a_1 + a_2 = 1, \quad C_1 := b_1 + b_2 + c = 1.$$
$$p \geqslant 2: \quad C_2 := -b_2 + c(b_3 - a_2) = \tfrac{1}{2}.$$
$$p \geqslant 3: \quad C_3 := \tfrac{1}{2}b_2 + \tfrac{1}{2}c(a_2 - 2b_3) = \tfrac{1}{6}, \quad -a_2 + 2b_3 + (b_3 - a_2)^2 = 0.$$

In the case of the LM method (2.2), the quantities $C_j - 1/j!$ represent the error constants of the method.

Furthermore, we shall need the characteristic polynomial of (2.1) which is given by

$$\zeta^2 - S(z)\zeta + P(z), \quad S(z) := 1 + (b_1 + ca_1)z, \quad P(z) := -z\left[b_2 + ca_2 + cb_3 z\right],$$

where z runs through the eigenvalues of the matrix $h\partial f/\partial y$. For a given real value of z, this polynomial has its roots on the unit disk if the Hurwitz inequalities $P(z) \leqslant 1$ and $|S(z)| \leqslant P(z) + 1$ are satisfied.

Finally, we remark that the storage requirements needed to implement (2.1) are reduced if we choose $a_2 = 0$. We observe that if b_1 also vanishes in (2.1), then the "two-processor" methods (2.1) can be implemented in such a way that (2.1) and (2.2) require the same amount of storage.

2.1. *First-order methods with increased real stability interval*

We start with first-order methods. Imposing the corresponding order conditions we find that, within the family (2.2), the real stability interval is determined by the Hurwitz inequalities $b_2 z \geqslant -1$ and $(1 - 2b_2)z \geqslant -2$. An elementary calculation reveals that for $b_2 = \frac{1}{4}$, that is, for the method

$$y_{n+1} = y_n + \tfrac{1}{4}h\left[3f_n + f_{n-1}\right], \tag{2.3}$$

the real stability interval is maximized and is given by $[-4, 0]$. Furthermore, the error constant is given by $C_2 - \frac{1}{2} = -\frac{3}{4}$. Thus, we are faced with the task to show that the family (2.1′) contains first-order methods with larger stability intervals and comparable error constants.

Theorem 1. *The two-parameter family of methods*

$$\begin{aligned} f_n &= f(y_n), \\ g_n &= f\left(a_1 y_n + (1 - a_1)y_{n-1} + \frac{1}{9c}hf_{n-1}\right), \quad c \neq 0, \\ y_{n+1} &= y_n + h\left[\left(\tfrac{1}{3} - ca_1\right)f_n + \left(\tfrac{2}{3} - c + ca_1\right)f_{n-1} + cg_n\right], \end{aligned} \tag{2.4}$$

is first-order accurate with error constant

$$C_2 - \tfrac{1}{2} = -\tfrac{19}{18},$$

and it possesses the largest possible real stability interval $[-6, 0]$, *for all values of* a_1 *and* c.

Proof. By imposing the first-order conditions we eliminate a_2 and b_2 from the polynomials $S(z)$ and $P(z)$ defined above. Let us write

$$S(z) = 1 + qz, \qquad P(z) = -z\left[1 - q + rz\right], \quad q := b_1 + ca_1, \quad r := cb_3,$$

showing that there are two independent parameters for maximizing the real stability interval. We shall simplify this optimization problem by imposing two assumptions which will be verified afterwards. The first assumption is $r > 0$. The second assumption is suggested by considering the Hurwitz conditions. Geometrically, the second Hurwitz condition $|S| \leqslant P + 1$ means that the graphs of the parabolas defined by $P(z) + 1$ and $-P(z) - 1$ have to enclose the graph of the line presented by S on the stability interval. This interval should be as large as possible while satisfying the second Hurwitz condition $P \leqslant 1$. Intuitively, the stability interval is maximal if $P(z)$ actually assumes the value 1. This second assumption leads us to the relation

$$r = \left(\frac{q-1}{2}\right)^2.$$

Let z_0 denote the point on the z-axis where the line S intersects either the parabola $P(z) + 1$ or the parabola $-P(z) - 1$. Then the stability boundary β equals $-z_0$, i.e.,

$$\beta = \beta(q) = 2 \min\left\{\frac{1}{(q-1)^2}, \frac{1 - 2q + \sqrt{6q^2 - 8q + 3}}{(q-1)^2}\right\}.$$

The value of $\beta(q)$ is maximized for $q = \frac{1}{3}$ resulting in $\beta = 6$ and error constant $-\frac{19}{18}$. Next we have to verify whether our two assumptions are correct. These assumptions have been confirmed by a numerical search in the two-parameter space spanned by q and r. □

From this theorem, it follows that the error constant C_2 is independent of the parameters a_1 and c, and therefore it is not possible to increase the accuracy by a judicious choice of these parameters. An alternative is to exploit the freedom of the two parameters for reducing the storage requirements. For example, by choosing $a_1 = 1$ and $c = \frac{1}{3}$, we obtain the method

$$\begin{aligned} f_n &= f(y_n), \\ g_n &= f(y_n + \tfrac{1}{3}hf_{n-1}), \\ y_{n+1} &= y_n + \tfrac{1}{3}h[2f_{n-1} + g_n]. \end{aligned} \tag{2.5}$$

2.2. Second-order methods with increased real stability interval

Next we turn to second-order methods. The family (2.2) contains just one second-order method and this method is the Adams–Bashforth method

$$y_{n+1} = y_n + \tfrac{1}{2}h[3f_n - f_{n-1}]. \tag{2.6}$$

The Adams–Bashforth method possesses the real stability interval $[-1, 0]$ and the error constant $-\frac{5}{12}$. We shall show that the family (2.1) contains a two-parameter family of second-order methods with negative stability interval $[-\frac{4}{3}, 0]$.

It is elementary verified that by imposing the conditions for second-order accuracy, the Hurwitz inequalities assume the form

$$cb_3z^2 + (cb_3 - \tfrac{1}{2})z + 1 \geqslant 0, \qquad cb_3z^2 + z \leqslant 0, \qquad cb_3z^2 + 2(cb_3 - 1)z - 2 \leqslant 0,$$

and that the largest interval of negative values of z satisfying these inequalities is obtained for $cb_3 = \frac{3}{4}$ and is given by $[-\frac{4}{3}, 0]$. Thus, choosing a_1 and c as free parameters, we have the result as formulated in the next theorem.

Theorem 2. *The two-parameter family of methods*

$$\begin{aligned} f_n &= f(y_n), \\ g_n &= f\Big(a_1 y_n + (1-a_1)y_{n-1} + \frac{3}{4c}hf_{n-1}\Big), \quad c \neq 0, \\ y_{n+1} &= y_n + h\big[\big(\tfrac{3}{4} - ca_1\big)f_n + \big(\tfrac{1}{4} - c + ca_1\big)f_{n-1} + cg_n\big], \end{aligned} \tag{2.7}$$

is second-order accurate and possesses the real stability interval $[-\frac{4}{3}, 0]$.

It is not possible to exploit the freedom of the two free parameters for raising the order of the method. However, by choosing these parameters such that the second third-order condition is satisfied, we can compute an error constant $C_3 - \frac{1}{6}$, so that we obtain a measure for the accuracy.

A simple calculation reveals that this condition is satisfied for

$$a_1 = \frac{1}{2}\left[1 - \frac{3}{2c} \pm \sqrt{1 - \frac{3}{c}}\right],$$

to obtain the error constant $C_3 - \frac{1}{6} = -\frac{19}{24}$ which is about twice as large as the error constant of (2.6).

As observed earlier, by choosing $a_1 = 1$, we reduce the storage requirements of the algorithm. Setting $c = \frac{3}{4}$, we obtain the simple method

$$\begin{aligned} f_n &= f(y_n), \\ g_n &= f(y_n + hf_{n-1}), \\ y_{n+1} &= y_n + \tfrac{1}{4}h\big[f_{n-1} + 3g_n\big]. \end{aligned} \tag{2.8}$$

2.3. Third-order methods

It is well known that zero-stable, explicit linear two-step methods cannot have order p greater than 2, so that methods of type (2.2) are at most second-order accurate. Next, we consider the attainable order of (2.1). From the order conditions given above it follows that for third-order accuracy five conditions are to be satisfied. Since there are six free parameters there is one parameter left. We shall choose c as the free parameter. It turns out that we cannot choose c such that the three additional conditions for fourth-order accuracy are satisfied. Thus, the attainable order of (2.1) is $p = 3$.

Unfortunately, the parameter c cannot be used for increasing the stability region of the method. This follows immediately from the Hurwitz inequalities used in the preceding subsection. These inequalities were derived under the condition of second-order accuracy leaving cb_3 as a free parameter. It can be shown that for third-order accuracy the value of cb_3 should equal $-\frac{5}{6}$, so that the Hurwitz inequalities are fixed for all third-order methods. This leads us to the following theorem.

Theorem 3. *The one-parameter family of methods*

$$\begin{aligned} f_n &= f(y_n), \quad a_1 := \frac{1}{2} + \frac{5}{6c} \pm \frac{1}{2}\sqrt{1 + \frac{10}{3c}}, \\ g_n &= f\Big(a_1 y_n + (1-a_1)y_{n-1} - \frac{5}{6c}hf_{n-1}\Big), \quad c \neq 0, \\ y_{n+1} &= y_n + h\big[\big(\tfrac{7}{3} - ca_1\big)f_n + \big(-\tfrac{4}{3} - c + ca_1\big)f_{n-1} + cg_n\big], \end{aligned} \tag{2.9}$$

is third-order accurate and possesses the real stability interval $[-\frac{1}{5}(11 - \sqrt{61}), 0]$.

As before, we consider the storage economic case where $a_1 = 1$. It is easily verified that this can be achieved by choosing $c = \frac{5}{12}$. The corresponding scheme is given by

$$\begin{aligned} f_n &= f(y_n), \\ g_n &= f(y_n - 2hf_{n-1}), \\ y_{n+1} &= y_n + h\left[\tfrac{23}{12}f_n - \tfrac{4}{3}f_{n-1} + \tfrac{5}{12}g_n\right]. \end{aligned} \tag{2.10}$$

3. *m*-processor algorithm

The algorithms described above can be generalized for use on m-processor computers. Consider the special explicit, multistep RK method

$$y_{n+1} = \sum_{j=1}^{k}\left[a_j y_{n+1-j} + h\sum_{i=1}^{m} b_{ij} f_{n+1-j,i}\right] + h\sum_{i=1}^{m} b_i f_{n+1,i}, \tag{3.1a}$$

where the right-hand side values $f_{n+1,i}$ are defined according to the formula

$$f_{n+1,s} := f\left(\sum_{j=1}^{k}\left[c_{sj} y_{n+1-j} + h\sum_{i=1}^{m} d_{sij} f_{n+1-j,i}\right]\right), \quad s = 1,\ldots,m. \tag{3.1b}$$

Evidently, the evaluation of the values $f_{n+1,1},\ldots,f_{n+1,m}$ can be done independently of each other. Thus, if m processors are available, then the required computation time for executing one step roughly corresponds to just one f-evaluation.

3.1. Linear multistep version

Let us introduce the $1\times m$-matrices $B_0 := (b_i)$ and $B_j := (b_{ij})$, the $m\times 1$-matrices $C_j := (c_{ij})$, the $m\times m$-matrices $D_j := (d_{sij})$, and the (column) m-vectors $\boldsymbol{f}_n := (f_{n,i})$, where $j = 1,\ldots,k$. Then the algorithm (3.1) can be written in the more compact form

$$y_{n+1} = \sum_{j=1}^{k}\left[a_j y_{n+1-j} + hB_j \boldsymbol{f}_{n+1-j}\right] + hB_0 \boldsymbol{f}_{n+1}, \tag{3.2a}$$

$$\boldsymbol{f}_{n+1} := f\left(\sum_{j=1}^{k}\left[C_j y_{n+1-j} + hD_j \boldsymbol{f}_{n+1-j}\right]\right), \tag{3.2b}$$

where, for any given vector $\boldsymbol{v} = (v_j)$, $f(\boldsymbol{v})$ denotes the vector with entries $f(v_j)$.

Suppose that $y_n,\ldots,y_{n+1-k}$ and $\boldsymbol{f}_n,\ldots,\boldsymbol{f}_{n+1-k}$ have already been computed, then (3.2) defines the computation of y_{n+1} and $\boldsymbol{f}_{n+1}$. Thus, (3.2) represents a k-step method for computing successively the vectors $(y_n, \boldsymbol{f}_n^{\mathrm{T}})$ for $n = k, k+1,\ldots$. By introducing the polynomials

$$\begin{aligned} \alpha(\zeta) &:= \zeta^k - a_1\zeta^{k-1} - a_2\zeta^{k-2} - \cdots - a_k, \\ \beta(\zeta) &:= B_0\zeta^k + B_1\zeta^{k-1} + B_2\zeta^{k-2} + \cdots + B_k, \\ \gamma(\zeta) &:= C_1\zeta^{k-1} + C_2\zeta^{k-2} + \cdots + C_k, \\ \delta(\zeta) &:= D_1\zeta^{k-1} + D_2\zeta^{k-2} + \cdots + D_k, \end{aligned}$$

the k-step method (3.2) can be presented in the linear multistep fashion

$$\alpha(E)E^{-k}y_n - h\beta(E)E^{-k}\boldsymbol{f}_n = 0, \quad \boldsymbol{f}_n := f\big(\gamma(E)E^{-k}y_n + h\delta(E)E^{-k}\boldsymbol{f}_n\big), \tag{3.3}$$

where E denotes the forward shift operator E defined by $Ey_n := y_{n+1}$. The resemblance with the conventional LM method

$$\rho(E)y_n - h\sigma(E)f_n = 0, \quad f_n = f(y_n),$$

is clear. Both methods employ, in addition to the recursion for the numerical solution y_n, an "auxiliary" recursion. In the linear multistep case this auxiliary recursion for f_n is sort of trivial, whereas in the case (3.3) the recursion for $\boldsymbol{f}_n$ is an essential part of the algorithm.

The method (2.1′) is a special ($k = 2$, $m = 2$) case of (3.3). Writing (2.1′) in the form

$$y_{n+1} - y_n - h\left[(b_2, 0)\begin{pmatrix} f_{n-1} \\ g_{n-1} \end{pmatrix} + (b_1, c)\begin{pmatrix} f_n \\ g_n \end{pmatrix}\right] = 0,$$

$$\begin{pmatrix} f_n \\ g_n \end{pmatrix} = f\left(\begin{pmatrix} 1 \\ a_1 \end{pmatrix} y_n + \begin{pmatrix} 0 \\ a_2 \end{pmatrix} y_{n-1} + h\begin{pmatrix} 0 & 0 \\ b_3 & 0 \end{pmatrix}\begin{pmatrix} f_{n-1} \\ g_{n-1} \end{pmatrix}\right),$$

we see that the polynomials α, β, γ and δ are given by

$$\alpha(\zeta) = \zeta^2 - \zeta, \qquad \beta(\zeta) = \big(b_1\zeta^2 + b_1\zeta,\ c\zeta^2\big),$$

$$\gamma(\zeta) = \begin{pmatrix} \zeta \\ a_1\zeta + a_2 \end{pmatrix}, \qquad \delta(\zeta) = \begin{pmatrix} 0 & 0 \\ b_3\zeta & 0 \end{pmatrix}.$$

3.2. Starting values

In order to start the recursion (3.2), we need k starting vectors $(y_n, \boldsymbol{f}_n^{\mathrm{T}})$. Let us try to approximate the vector $\boldsymbol{f}_n$ by means of y-values only. From (3.3) we deduce

$$\begin{aligned}
\boldsymbol{f}_n &:= f\big(\gamma(E)E^{-k}y_n + h\delta(E)E^{-k}\boldsymbol{f}_n\big) \\
&= f\big(\gamma(E)E^{-k}y_n + h\delta(E)E^{-k}f\big(\gamma(E)E^{-k}y_n + h\delta(E)E^{-k}\boldsymbol{f}_n\big)\big) \\
&= f\big(\gamma(E)E^{-k}y_n + h\delta(E)E^{-k}f\big(\gamma(E)E^{-k}y_n + h\delta(E)E^{-k}f\big(\gamma(E)E^{-k}y_n \\
&\qquad + h\delta(E)E^{-k}\boldsymbol{f}_n\big)\big)\big) = \ldots,
\end{aligned}$$

from which it follows that

$$\begin{aligned}
\boldsymbol{f}_n &= f\big(\gamma(E)E^{-k}y_n\big) + \mathrm{O}(h), \\
\boldsymbol{f}_n &= f\big(\gamma(E)E^{-k}y_n + h\delta(E)E^{-k}f\big(\gamma(E)E^{-k}y_n\big)\big) + \mathrm{O}(h^2), \\
\boldsymbol{f}_n &= f\big(\gamma(E)E^{-k}y_n + h\delta(E)E^{-k}f\big(\gamma(E)E^{-k}y_n + h\delta(E)E^{-k}f\big(\gamma(E)E^{-k}y_n\big)\big)\big) \\
&\qquad + \mathrm{O}(h^3), \ldots .
\end{aligned}$$

Evidently, the number of y-values needed to approximate $\boldsymbol{f}_n$ can be reduced by choosing zero matrices for C_k, $C_{k-1}, \ldots$, and D_k, $D_{k-1}, \ldots$. For instance, if all matrices C_j and D_j vanish except for C_1 and D_1, then

$$\begin{aligned}
\boldsymbol{f}_n &= f(C_1y_{n-1}) + \mathrm{O}(h), \\
\boldsymbol{f}_n &= f(C_1y_{n-1} + hD_1f(C_1y_{n-2})) + \mathrm{O}(h^2), \\
\boldsymbol{f}_n &= f(C_1y_{n-1} + hD_1f(C_1y_{n-2} + hD_1f(C_1y_{n-3}))) + \mathrm{O}(h^3), \ldots .
\end{aligned}$$

In this way, a pth-order approximation to f_n can be obtained by means of the values $y_{n-1}, \ldots, y_{n-p}$. As for every linear multistep method, these starting values are to be obtained by some self-starting method (e.g., a one-step method).

An alternative way of computing starting vectors f_n is possible in the case of strictly lower triangular matrices D_j. In that case, we deduce from (3.1b) that f_n can directly be expressed in terms of y-values:

$$f_{n,1} := f\left(\sum_{j=1}^{k} c_{1j} y_{n-j} \right), \qquad f_{n,2} := f\left(\sum_{j=1}^{k} \left[c_{2j} y_{n-j} + h d_{21j} f_{n-j,1} \right] \right),$$

$$f_{n,3} := f\left(\sum_{j=1}^{k} \left[c_{3j} y_{n-j} + h d_{31j} f_{n-j,1} + h d_{32j} f_{n-j,2} \right] \right), \ldots .$$

4. Accuracy and stability

4.1. Accuracy

The order of accuracy of the method (3.1) is said to be p if the residue left on substitution of the exact solution $y(t)$ into (3.1) is of order h^{p+1}. We shall indicate how the order equations can be obtained in terms of the polynomials α, β, γ and δ. Using the representation (3.3), we can write the order condition in the form

$$\alpha(E) y(t_n) - h\beta(E) f\big(\gamma(E) E^{-k} y(t_n) + h\delta(E) E^{-k} f\big(\gamma(E) E^{-k} y(t_n) + \cdots\big)\big) = O(h^{p+1}). \tag{4.1}$$

Assuming that $y(t)$ is sufficiently differentiable, we have $E = \exp(h\, \mathrm{d}/\mathrm{d}t)$; hence, by using the abbreviations

$$\alpha(E) = \alpha\left(\exp\left(h \frac{\mathrm{d}}{\mathrm{d}t}\right)\right) = a\left(h \frac{\mathrm{d}}{\mathrm{d}t}\right), \qquad \beta(E) = b\left(h \frac{\mathrm{d}}{\mathrm{d}t}\right),$$

$$\gamma(E) = c\left(h \frac{\mathrm{d}}{\mathrm{d}t}\right), \qquad \delta(E) = d\left(h \frac{\mathrm{d}}{\mathrm{d}t}\right),$$

and putting $c(0) = e := (1, 1, \ldots, 1)^{\mathrm{T}}$, we can expand (4.1) in powers of h. For instance,

$$a(0) y(t_n) + h\left[a'(0) - b(0) e\right] y'(t_n) + h^2 \left[\tfrac{1}{2} a''(0) - b(0)\left(c'(0) + d(0) e - k e\right) - b'(0) e\right] y''(t_n) + O(h^3) = O(h^{p+1}). \tag{4.1'}$$

By expressing the various derivatives of the functions a, b, c and d again in terms of the polynomials α, β, γ and δ we finally obtain the order equations. For example, on substitution of $a(0) = \alpha(1)$, $a'(0) = \alpha'(1)$, $a''(0) = \alpha'(1) + \alpha''(1)$ into (4.1'), and similar expressions for the other coefficient functions, we find the order equations:

$$p \geqslant 1: \quad \gamma(1) = e, \qquad \alpha(1) = 0, \qquad \alpha'(1) - \beta(1) e = 0.$$

$$p \geqslant 2: \quad \tfrac{1}{2}\left[\alpha'(1) + \alpha''(1)\right] - \beta(1)\left[\gamma'(1) + \delta(1) e - k e\right] - \beta'(1) e = 0.$$

4.2. Stability

Next we derive the linear stability condition for the method (3.1), that is, we apply the method to the linear test equation $y' = \lambda y$. Again using the representation (3.3), we obtain the recursion

$$\alpha(E)y_n - \beta(E)hf_n = 0, \qquad \lambda h\gamma(E)y_n - \left[E^k - \lambda h\delta(E)\right]hf_n = \mathbf{0}, \tag{4.2}$$

or equivalently,

$$\begin{pmatrix} \alpha(E) & -\beta(E) \\ \lambda h\gamma(E) & -E^k I + \lambda h\delta(E) \end{pmatrix}\begin{pmatrix} y_n \\ hf_n \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{0} \end{pmatrix}. \tag{4.2'}$$

We now use the following lemma (cf. [1, p.428]):

Lemma 4. *Let the sequence of vectors* $\{v_n\}$ *satisfy the difference equation* $G(E)v_n = \mathbf{0}$, *where the entries of the constant matrix* $G(\zeta)$ *are polynomials in* ζ. *Then each component of* v_n *satisfies again a homogeneous difference equation with characteristic polynomial* $\det[G(\zeta)]$.

Application of this lemma to (4.2') reveals that y_n satisfies a difference equation with characteristic polynomial

$$C(\zeta;\ \lambda h) := \det\begin{pmatrix} \alpha(\zeta) & -\beta(\zeta) \\ \lambda h\gamma(\zeta) & -\zeta^k I + \lambda h\delta(\zeta) \end{pmatrix}. \tag{4.3}$$

Following the linear stability theory for multistep methods, we define the stability region S by the set of points in the complex z-plane where the polynomial $C(\zeta;\ z)$ has its roots on the unit disk, and require that λh lies in S when λ runs through the eigenvalues of the matrix $\partial f/\partial y$.

5. Numerical experiments

Of the various two-step methods discussed in the preceding sections, we compare methods where all free parameters are used for maximizing the real stability interval and methods where these parameters maximize the order of accuracy. The main characteristics of these methods are once again listed in Table 1. The methods (2.3) and (2.6) are "one-processor" algorithms, and (2.5) and (2.10) are "two-processor" algorithms. The methods were applied to test problems taken from [2] and [5], and are specified in Tables 2 and 3. The maximum absolute errors produced at the end point $t = T$ are denoted by e_1 for the "one-processor" algorithms and by e_2 for the "two-processor" algorithms. Asterisks indicate development of instabilities. Table 2

Table 1
Methods used in the experiments

Method	(2.3)	(2.5)	(2.6)	(2.10)
Order	1	1	2	3
Stability interval	$[-4, 0]$	$[-5.8, 0]$	$[-1, 0]$	$[-0.64, 0]$
Starting values	y_0, y_1	y_0, y_1	y_0, y_1	y_0, y_1

Table 2
Stability test for Problem D1 [2]: $y_1' = 0.2(y_2 - y_1)$, $y_2' = 10y_1 - (60 - \frac{1}{8}y_3)y_2 + \frac{1}{8}y_3$, $y_3' = 1$, $y_1(0) = 0$, $y_2(0) = 0$, $y_3(0) = 0$, $T = 400$

	$h^{-1}=8$	$h^{-1}=10$	$h^{-1}=12$	$h^{-1}=14$	$h^{-1}=16$	$h^{-1}=18$
(2.3): $e_1 \approx$	*	*	*	*	$10^{-1.2}$	$10^{-1.2}$
(2.5): $e_2 \approx$	*	$10^{-0.83}$	$10^{-0.91}$	$10^{-1.0}$	$10^{-1.0}$	$10^{-1.1}$

Table 3
Accuracy test for Problem B5 [5]: $y_1' = y_2y_3$, $y_2' = -y_1y_3$, $y_3' = -0.51y_1y_2$, $y_1(0) = 0$, $y_2(0) = 1$, $y_3(0) = 1$, $T = 20$

	$h^{-1}=32$	$h^{-1}=64$	$h^{-1}=128$	$h^{-1}=256$
(2.6): $e_1 \approx$	$10^{-2.5}$	$10^{-3.2}$	$10^{-3.8}$	$10^{-4.4}$
(2.10): $e_1/e_2 \approx$	11	22	43	84

presents results for the first-order methods (2.3) and (2.5) showing the improved stability of the "two-processor" method. Table 3 presents similar results for the second-order Adams–Bashforth method (2.6) and the third-order "two-processor" method (2.10). We recall that on two-processor computers, all methods require one right-hand side evaluation per step.

Acknowledgement

We are grateful to the referee for drawing our attention to a deficiency in the original version of Theorem 1 which resulted in a larger stability boundary than found initially.

References

[1] H. Brunner and P.J. van der Houwen, *The Numerical Solution of Volterra Equations*, CWI Monograph No. 3 (North-Holland, Amsterdam, 1986).
[2] W.H. Enright, T.E. Hull and B. Lindberg, Comparing numerical methods for stiff systems of ODEs, *BIT* **15** (1975) 10–48.
[3] E. Hairer, S.P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I. Nonstiff Problems* (Springer, Berlin, 1987).
[4] P.J. van der Houwen, *Construction of Integration Formulas for Initial Value Problems* (North-Holland, Amsterdam, 1977).
[5] T.E. Hull, W.H. Enright, B.M. Fellen and A.E. Sedgwick, Comparing numerical methods for ordinary differential equations, *SIAM J. Numer. Anal.* **9** (1972) 603–637.